第一章

網路安全評估概述

本章將介紹網路攻擊和防禦背後的經濟理論，以及當前的事態發展走向，要建立具有相當防禦能力的環境，必須主動積極朝安全防護前進，首先就是透過評估手段，了解自己暴露了多少弱點，從程式碼靜態分析，到執行系統動態測試，有許多的評估方法，這裡會對測試選項進行分類，並說明本書所涵蓋的領域。

技術發展

本書第一版大約在 20 年前撰寫，由於政府機構及犯罪組織的競賽，促成網路漏洞產業蓬勃發展，當時，零時差利用的商業模式尚未出現，都是透過網路聊天室（IRC）進行**零時差交易**，而駭客是網路生態的頂層掠食者。

如今，態勢相當令人憂心，目前的生活形態極度依賴電腦網路和程式，這些轉變相當複雜，且朝多個方向快速發展（想想雲端應用、連網的醫療設備和自動駕駛），愈來愈多消費者使用有缺陷的產品，安全漏洞的數目也與日俱增。

網際網路是全球經濟體系的主要推動者，幾乎所有活動都與它息息相關，**國際應用系統分析研究所**（IIASA）預測，若一個國家的網際網路服務完全癱瘓，三天之內就會造成食品供應鏈無法運作[1]。

1 Leena Ilmola-Sheppard 和 John Casti 合撰的《Case Study: Seven Shocks and Finland》（http://pure.iiasa.ac.at/10111/），發表在 *Innovation and Supply Chain Management Vol 7*（創新與供應鏈管理第 7 卷）2013 年第 3 號的 112 到 124 頁。

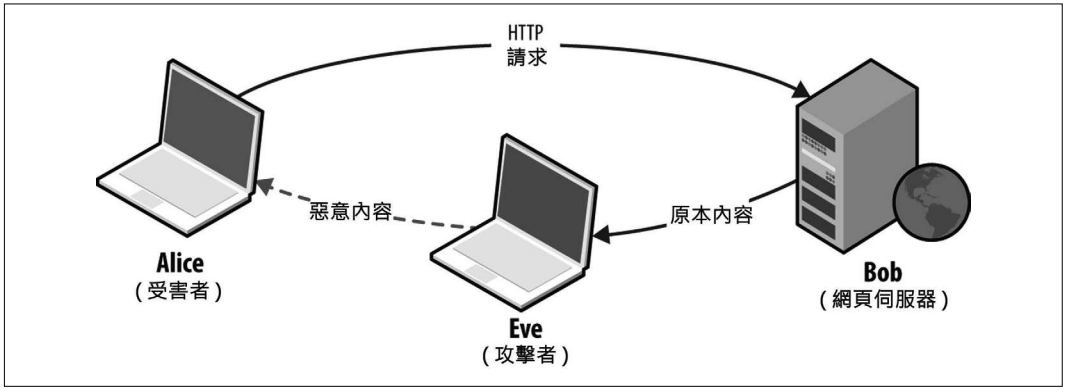

圖 1-1：利用網路傳送惡意內容

中間人攻擊並不限於未加密的網路連線，有謀略的攻擊者，可透過軟體的弱點取得私有金鑰內容，或直接利用作業系統的安全缺陷，仍然可能成功攻擊有效加密的網路連線（如 HTTPS）。

攻擊伺服器端軟體

由於日益增加的抽象層和新興技術，伺服器軟體不見得就比較好，Rails 應用伺服器和 Nginx 的反向代理功能在 2013 年都遭受到嚴重的程碼執行弱點攻擊：

- Rails 2.3 和 3.x 的 Action Pack YAML 解序列化弱點 [11]。
- Nginx 1.3.9 到 1.4.0 的區塊式編碼堆疊溢位漏洞 [12]。

Nginx 的區塊式編碼漏洞和 Neel Mehta 在 2002 年從 Apache HTTP 伺服器找到的弱點 [13] 相似，這顯示新軟體的開發人員並未汲取前人失敗的教訓，才會重蹈覆轍。

11 參閱 CVE-2013-0156（http://bit.ly/2aNz83C）。
12 參閱 CVE-2013-2028（http://bit.ly/2aNzKpZ）。
13 參閱 CVE-2002-0392（http://bit.ly/2aNzxTP）。

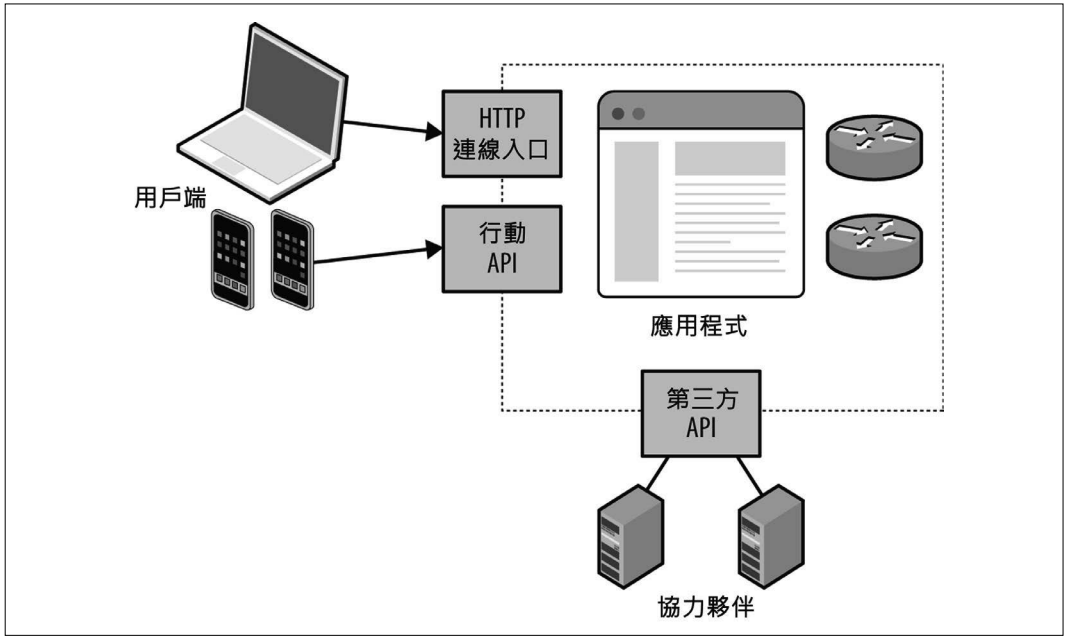

圖 1-2：應用程式使用的網頁服務

網路社交工程

筆者的滲透測試生涯中，某些行動是透過網路社交工程手法，底下提供兩種攻擊情境：

- 建置一組網頁伺服器，偽裝成合法的資源，然後寄送一份引人矚目的電子郵件給使用者，但裡頭嵌有惡意網頁的超鏈結。
- 仿冒可信任的來源，例如朋友或同事，用電子郵件、即時訊息或其他方式，直接將惡意檔案（像是利用 Excel 或 Adobe Acrobat 做成的檔案）送到使用者手上。

最近接受一家金融機構委託，執行魚叉式網路釣魚行動，利用偽造的 SSL VPN 端點，並發送郵件給 200 名使用者，指示他們登入公司新建置的 VPN 閘道。兩小時之內就有 13 位使用者輸入他們的 AD 帳號、密碼及雙因子驗證的憑據，第 9 章會詳細介紹釣魚的手法和工具。

弱點掃描

依照找到的 IP 區段，攻擊者接著執行大規模掃描，確認可用的網路服務，做為後續特定的攻擊目標，可能利用它們進行程式碼執行攻擊、取得外洩資訊或阻斷服務，網路掃描工具可以辨識服務特徵、探測及檢驗已知的問題，常見的掃描工具有 Nmap、Nessus、Nexpose 和 QualysGuard。

利用弱點掃描收集到的資訊包括已曝光的網路服務和週邊資訊，例如伺服器回應的 ICMP 訊息及防火牆的 ACL 設定等，掃描工具也會回報已知弱點及回應訊息，這些資訊將進一步應用在調查階段。

漏洞調查

有時能從網際網路上公開的郵遞論壇和討論區找到軟體漏洞，但愈來愈多漏洞是賣給像零時差懸賞計畫（ZDI）這類私人機構，再依其政策將問題回報給開發商及付費的會員，依據 Immunity 公司調查顯示，平均要 348 天，開發商才會完成軟體零時差弱點修正。

有些經銷商或中介商並不會將漏洞通知軟體供應商，而是提供漏洞利用方法給商業客戶，在負責任的披露、濫用零時差漏洞和公開討論之間，漏洞資訊並不一致，兩種漏洞擴散途徑如圖 2-1 所描述。

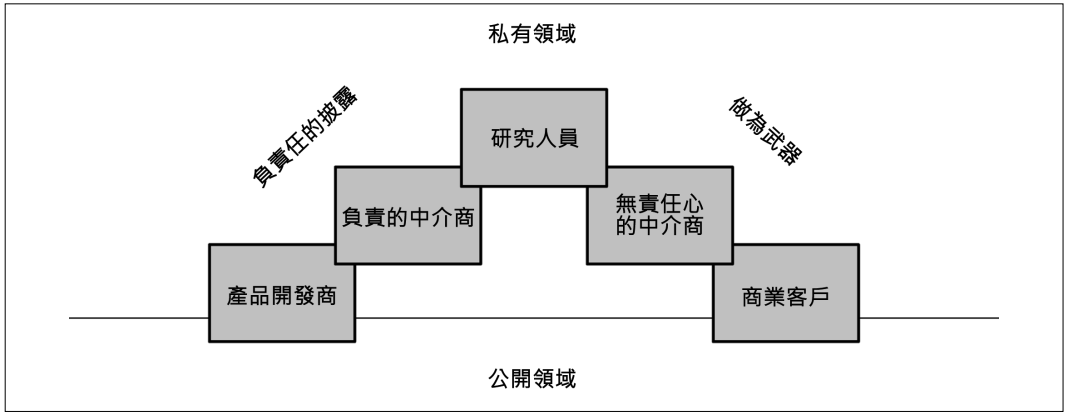

圖 2-1：常見的漏洞資訊擴散途徑

以往為了提升執行速度，程式碼可能自我改寫，現在多數處理器都能針對唯讀程式碼進行最佳化，自我改寫反而會造成性能下降，因此可大膽假設，程序嘗試修改內文區段的內容是不必要的。

像 Java 和微軟 .NET 的即時（JIT）編譯器會預備可執行碼所需的記憶體分頁，再以指令填充之，因此，應用程式通常在內文區段之外修改後再寫到分頁中。

資料和 BSS 區段

資料和 BSS 區段包含程式使用的靜態變數和全域變數，這些記憶體區段通常可供讀寫，有時在此區段內的指令也可以被執行。

堆積記憶體

堆積記憶體通常是程式大塊分配的記憶體，應用程式使用堆積記憶體保存函式返回後仍需持續使用的資料，系統使用配置及釋放函式管理堆積記憶體上的資料，C 常用 *malloc* 配置大塊記憶體，欲回收時則呼叫 *free* 函式，當然還可以使用其他最佳化的記憶體配置函式。

不同的作業系統使用不同的演算法來管理堆積記憶體，表 3-1 是不同平臺上的實作方式。

表 3-1：堆積記憶體的管理演算法

實作方式	作業系統
GNU libc（Doug Lea）	Linux
AT&T System V	Solaris、IRIX
BSD（Poul-Henning Kamp）	FreeBSD、OpenBSD、Apple OS X
BSD（Chris Kingsley）	4.4BSD、Ultrix、某些版本的 AIX
Yorktown	AIX
RtlHeap	Windows

```
remarks:        http://www.nic.ad.jp/en/db/whois/en-gateway.html or
remarks:        whois.nic.ad.jp for WHOIS client. (The WHOIS client
remarks:        defaults to Japanese output, use the /e switch for English
remarks:        output)
changed:        apnic-ftp@nic.ad.jp 20050729
source:         JPNIC
```

OS X 用戶端的 *whois* 程式，用「-a」查詢 ARIN 伺服器、「–A」查詢 APNIC，有些版本的 whois 則可任意指定查詢的伺服器，例如 *whois nintendo -h whois.apnic.net*，更複雜的，每個伺服器支援的語法還不盡相同，必要時請參考 whois 及其配對使用的伺服器之說明文件。

使用 WHOIS 的網頁界面

除了命令列工具外，也可以利用註冊機構提供的查詢網頁來取得資訊，圖 4-11 是利用受測標的所在地的郵遞區號列舉註冊在 RIPE 的相關 IP 資料。

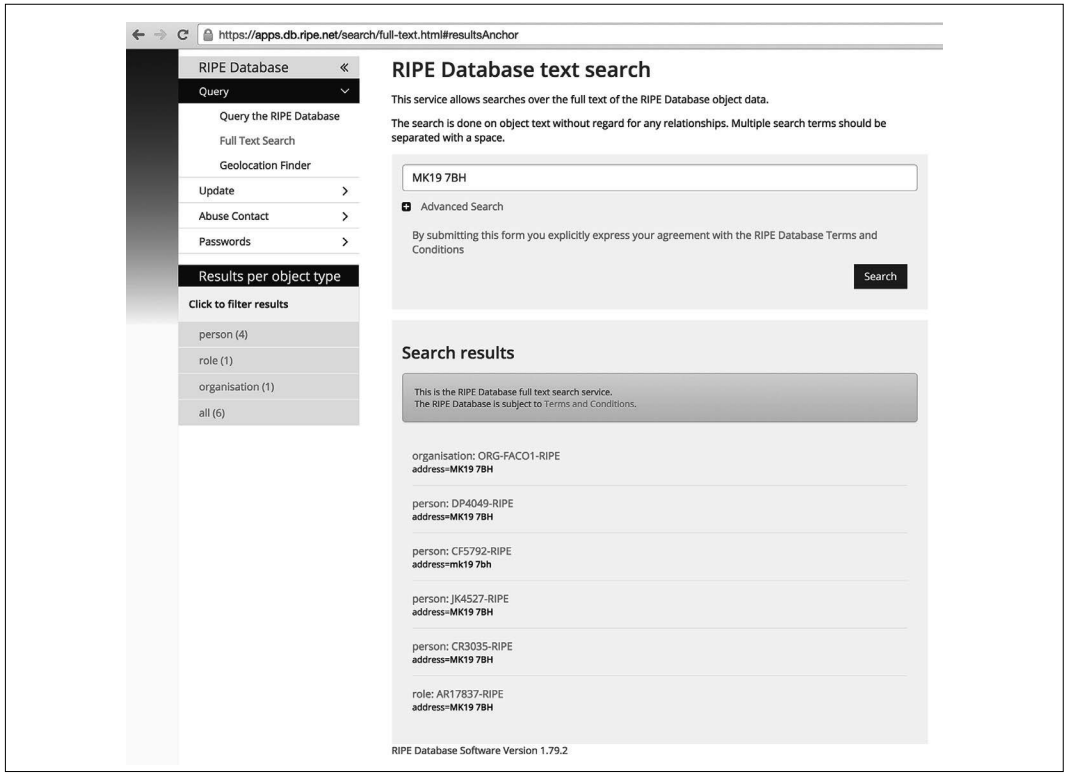

圖 4-11：利用郵遞區號向 RIPE 查詢

BGP 列舉

在網際網路上往來的封包是由邊界閘道協定（BGP）和自治系統（AS）編號所控制，網際網路編號分配機構（IANA）分派 AS 編號給 RIR，RIR 再分配給網際網路服務供應商（ISP）及其他組織，所以它們可以管理自己的 IP 網路路由，並與上游連線。

範例 4-3 的 WHOIS 查詢顯示任天堂的 AS 編號：

```
Nintendo Of America inc. (AS11278) NINTENDO 11278
```

因此可與 HE BGP 的查詢結果交叉比對，找出由此 AS 編號所通告的 IPv4 前綴位址，結果如圖 4-12 所示，同樣方式，亦可列舉 AS 通告的 IPv6 位址空間。

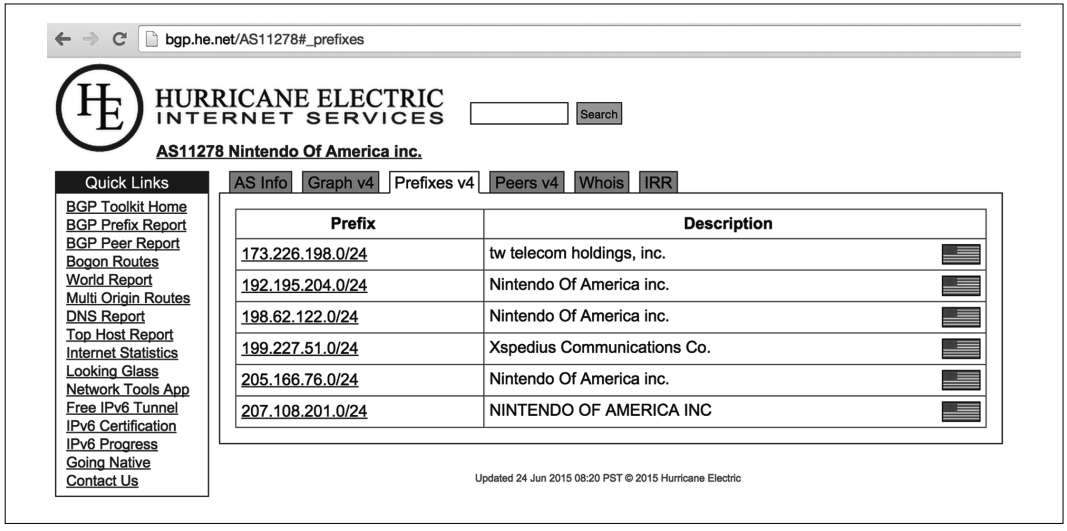

圖 4-12：交叉參照 AS 編號與 IP 區段

DNS 查詢

nslookup 和 *dig* 命令列工具可以查詢名稱伺服器，亦有自動化工具可以進行反向解析遍掃及正向解析猜解。

表 4-5 是 DNS 資源紀錄欄位簡要說明，除了 AAAA 的 IPv6 位址和 SRV 服務定位器紀錄外，在 RFC 1035 有全部紀錄的底層機制和使用的詳細說明。

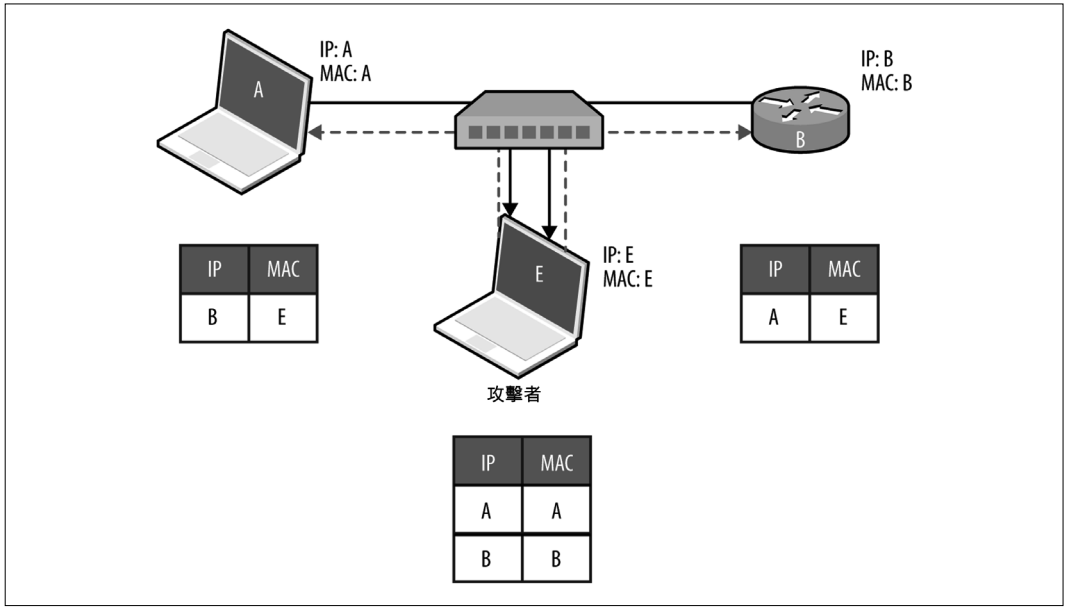

圖 5-3：ARP 快取毒化

要控制通訊兩方的流量（例如主機和區域閘道器間），首先要啟用攻擊者系統上的 IP 封包轉遞（IP forward）功能，並主動將偽造的 ARP 回應封包送給欲毒化的兩方系統，Cain & Abel 和 Ettercap[3] 等工具可以自動處理這些事情，在進行中間人攻擊時，駭客可利用一些手法從中截取資料和機密，包括下列這些：

- 使用 *sslstrip*[4] 將 HTTPS 連線降級為 HTTP。
- *easy-creds*[5] 可從 Ettercap、*ssltrip* 及其他工具的產物收集身分憑據資料。
- 利用 Laurent Gaffie 的 Responder[6] 回應名稱解析請求。
- 利用服務角色模仿的技倆，提供 Evilgrade[7] 的惡意內容給受害人。
- 使用 Metasploit 提供模仿的服務，執行進一步攻擊。

3 參考 GitHub 上的 *Ettercap*（https://ettercap.github.io/ettercap/）。
4 參考 *sslstrip* 開發者 Moxie Marlinspike 的網站（https://moxie.org/software/sslstrip/）。
5 參考 GitHub 上的 *easy-creds*（https://github.com/brav0hax/easy-creds）。
6 參考 GitHub 上的 *Responder*（https://github.com/lgandx/Responder）。
7 參考 GitHub 上的 *Evilgrade*（https://github.com/infobyte/evilgrade）。

網路上有一支使用 Scapy CDP 程式庫的 Python 參考腳本[20]，只要移除 *while* 迴圈，並且加入 *CDPMsgVoIPVLANReply*（type 15）的欄位就可篡改 CDP 訊框內容，對思科的網路電話或其他系統進行攻擊。

802.1D STP

生成樹協定（STP）是用來防止網路拓樸形成環路的機制，交換器使用網路橋接協定資料單元（BPDU）自我調整，將多餘的連接埠之狀態設為阻斷（*blocking*），以便切斷環路。圖 5-11 是典型的 STP 組態，表 5-5 說明四種連接埠狀態。

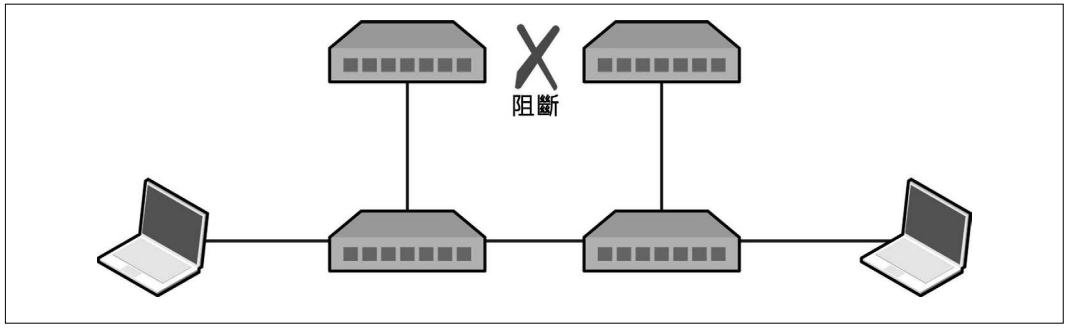

圖 5-11：使用 STP 的 802.3 乙太網路

表 5-5：在 STP 網路的連接埠狀態

狀態	說明
Disabled（停用）	關閉電氣信號，直到啟用為止
Blocking（阻斷）	除了 BPDU 訊框外，忽略其他所有訊框
Listening（監聽）	交換器監聽 BPDU 訊框，以便建立無環路的網路樹
Learning（學習）	交換器利用訊框的來源 MAC 位址建立轉送表
Forwarding（轉送）	連接埠完全運作，轉送所有進入或離開此交換器的訊框

20 參考 *http://examples.oreilly.com/networksa/tools/cdp_flooder.py*。

在選舉根橋接器時，每臺交換器選擇自己的**根埠**和**指定埠**，根埠提供到達根橋接器的最佳路徑，剩下的就是指定埠，有關 BPDU 的選舉過程及連接埠的設定，可參考 Kevin Lauerman 和 Jeff King 的報告[21]，裡頭有詳細運作的範例和拓樸圖。

監視 BPDU 訊框

如範例 5-10 所示，利用 Yersinia 的 STP 協定功能，可以擷取及顯示 BPDU 訊框內容，如果網路卡不能擷取 BPDU 訊框，就無法成功攻擊 STP。

範例 5-10：用 Yersinia 顯示 BPDU 訊框

```
┌─── yersinia 0.7.3 by Slay & tomac - STP mode ──────────────── [10:29:40] ┐
│     RootId            BridgeId          Port     Iface Last seen          │
│     5080.760F0E13AC58 CB09.E7CD90117CAA 8002     eth1 26 Aug 10:29:39     │
│     5080.760F0E14AC58 CB09.E7CD90127CAA 8002     eth2 26 Aug 10:29:38     │
│     5080.760F0E13AC58 CB09.E7CD90117CAA 8002     eth2 26 Aug 10:27:05     │
│     5080.760F0E14AC58 CB09.E7CD90127CAA 8002     eth1 26 Aug 10:26:59     │
│                                                                           │
```

接管根橋接器

連線到有兩臺交換器的環境上，使用 Ettercap 建立橋接，並用 Yersinia 發送特製的 BPDU 訊框到每個網路卡，圖 5-12 是最終的網路拓樸，可看到交換器上的流量都因流向攻擊者電腦而洩露。

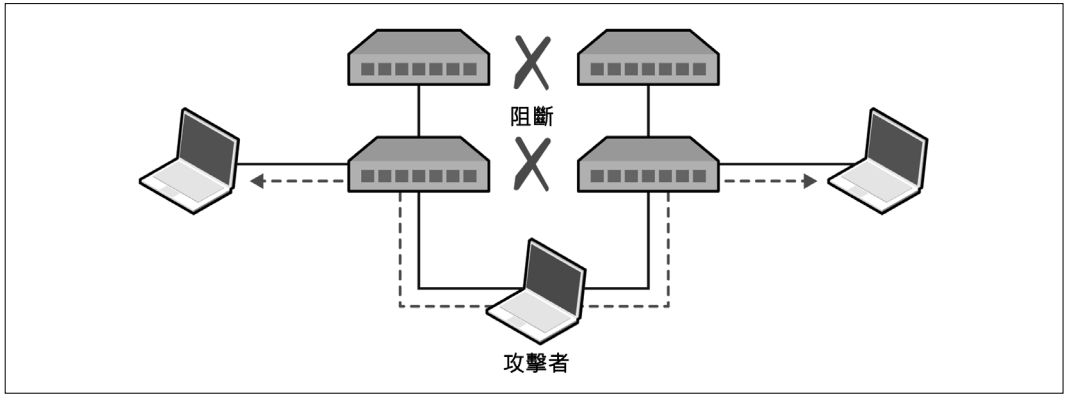

圖 5-12：接管 STP 的根橋接器

21 Kevin Lauerman 和 Jeff King 於 2010 年為思科而寫的報告：「STP MiTM Attack and L2 Mitigation Techniques」（http://bit.ly/2aNTgD3）。

表 5-6：DHCP 訊息的類型

訊息	發送端	說明
DHCPDISCOVER	用戶端	發出廣播封包徵求 DHCP，並選擇性附上最近一次使用的 IP 位址，向網路申請 IP 租賃
DHCPOFFER	伺服器	收到用戶端的徵求後，伺服器先預留一組 IP，然後回應含有此 IP 位址、子網路遮罩及 DHCP 選項（例如 DNS 伺服器及路由器明細）等的提議資料
DHCPREQUEST	用戶端	用戶端會收到來自多臺伺服器（若有）的 DHCP 提議資料，用戶端使用此提議訊息正式向特定的伺服器請求 IP 位址
DHCPACK	伺服器	伺服器的最終確認訊息，內容包括租賃期限及其他 DHCP 的選用資訊（如 WPAD 伺服器）
DHCPNAK	伺服器	通知用戶端網路位址資訊不正確，例如用戶變更子網路或者租約已到期
DHCPDECLINE	用戶端	表示用戶端一直使用此網路位址
DHCPRELEASE	用戶端	取消租約並放棄網路位址
DHCPINFORM	用戶端	請求其他網路設定參數（已經設置本機 IP 位址及子網路情形下）

一種主動攻擊 DHCP 的手法是**設定偽冒的伺服器**，通常合併使用泛洪方式阻斷合法 DHCP 伺服器的服務，其效果就是由惡意的預設閘道器、DNS 伺服器及 WPAD 提供服務給用戶端。

識別 DHCP 伺服器及其組態

範例 5-12 展示如何使用 Nmap 廣播探索訊息來列舉 DHCP 伺服器[23]，這種方式也可用在 DHCPv6 上[24]，由於此協定是以 UDP 方式運作，為求謹慎，建議多做幾次。

範例 5-12：使用 Nmap 識別 DHCP 伺服器

```
root@kali:~# nmap --script broadcast-dhcp-discover

Starting Nmap 6.49BETA4 (https://nmap.org) at 2015-08-26 07:31 EDT
Pre-scan script results:
| broadcast-dhcp-discover:
|   Response 1 of 1:
|     IP Offered: 192.168.1.5
```

23 參考 Nmap 的 *broadcast-dhcp-discover* 腳本（http://bit.ly/2aNTBWh）。

24 參考 Nmap 的 *broadcast-dhcp6-discover* 腳本（http://bit.ly/2aNTtGd）。

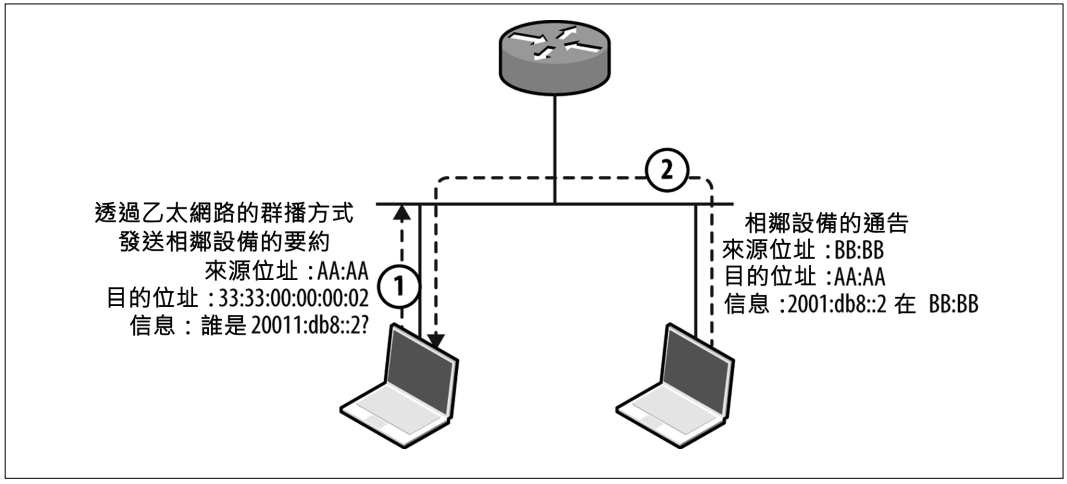

圖 5-19：IPv6 相鄰設定的要約與通告

列舉區域內的 IPv6 主機

可用利用 THC IPv6[54] 套件及 Metasploit 模組來識別區域內的 IPv6 主機，範例 5-21 是在 Kali 利用 ping6 廣播一組 ICMPv6 的 echo 訊息到 ff02::1（IPv6 的群播位址），然後由 *eth1* 接聽活動中的主機回應，之後可如範例 5-22 所示，使用 Nmap 掃描這些主機。其中 fe80::/10 是區域鏈路的 IPv6 前綴位址。

範例 5-21：使用 ping6 識別 IPv6 的相鄰主機

```
root@kali:~# ping6 -c2 -I eth1 ff02::1
PING ff02::1(ff02::1) from fe80::20e:c6ff:fef0:2965 eth1: 56 data bytes
64 bytes from fe80::20e:c6ff:fef0:2965: icmp_seq=1 ttl=64 time=0.040 ms
64 bytes from fe80::1a03:73ff:fe27:35a8: icmp_seq=1 ttl=64 time=1.23 ms
64 bytes from fe80::217:f2ff:fe0f:5d19: icmp_seq=1 ttl=64 time=1.23 ms
64 bytes from fe80::426c:8fff:fe2a:e708: icmp_seq=1 ttl=64 time=1.23 ms
64 bytes from fe80::e4d:e9ff:fec5:8f53: icmp_seq=1 ttl=64 time=1.47 ms
64 bytes from fe80::3ed9:2bff:fe9f:bc94: icmp_seq=1 ttl=64 time=1.62 ms
64 bytes from fe80::ba2a:72ff:fef1:b747: icmp_seq=1 ttl=64 time=1.85 ms
64 bytes from fe80::a2d3:c1ff:fed1:2a8e: icmp_seq=1 ttl=64 time=2.18 ms
```

54 參考 https://github.com/vanhauser-thc/thc-ipv6。